REVISED

TECHNICAL MEMORANDUM (NASA) 23

A PROPOSED MICROCOMPUTER IMPLEMENTATION OF AN OMEGA NAVIGATION PROCESSOR

Documentation of current status of research pertaining to a microprocessor-based Omega navigation processor to be used in conjunction with the Ohio University Avionics Engineering Center Omega sensor processor is presented.

by

John D. Abel
Avionics Engineering Center
Department of Electrical Engineering
Ohio University
Athens, Ohio 45701

March 1976

Supported by

National Aeronautics and Space Administration
Langley Research Center
Langley Field, Virginia
Grant NGR 36-009-017

I. OBJECTIVES

The basic objective of this project is to utilize the digital outputs of the Ohio University Avionics Engineering Center OMEGA sensor processor to derive pilot-usable navigation information. Some of the important design parameters are: cost, size, accuracy, minimal pilot input and simple, usable outputs.

II. CONCEPTUAL DESIGN

Many system configurations are possible; therefore, extensive literature research was conducted. Following suggestions made at NASA Joint-University Program Quarterly Conferences, we decided to develop a microprocessor-based navigation system. This gives flexibility in a small package and at a reasonable cost. Figure 1 illustrates a concept block diagram. This paper deals with the Omega process only; later work will include air-data processing.

The system must be able to accept data from the sensor processor and from pilot input controls. It must also output information to a pilot indicator and an incremental recorder (for experimental monitoring). To achieve maximum accuracy, skywave correction of the incoming signal is necessary. Considerable effort was made to investigate various means of implementing the skywave corrections. All of the methods studied place practical constraints on range, time of validity, finite accuracy, data memory size and cost, and real-time processing constraints. Storage for four stations is desired. Tradeoffs can be made with regard to range (several states to 1/2 U.S.) and time of validity (two months to one year). A brief outline of several methods follows:

Tabular Storage - Use read-only memory (ROM) to store existing precalculated skywave corrections[1] as a function of position (within 40 x 40) and time (day, hour). This makes softwave implementation simple, but requires large ROM capacity.

Compressed Tabular Storage - Enables storage of more information per equivalent ROM by utilizing the characteristic near-constancy of the skywave correction curve except at the sunrise and sunset transitions.

Polynomial Approximations - This technique involves a time-independent, distance-dependent quadratic polynomial to define a two-dimensional polynomial surface approximation of the skywave corrections[2] [$f(x,y) = a_1 + a_2x + a_3y + a_4xy + a_5x^2 + a_by^2$]. To achieve relative time independence a new set of constants must be used for each of the 24 hours.

This method saves some ROM storage, but increases software complexity and processing time.

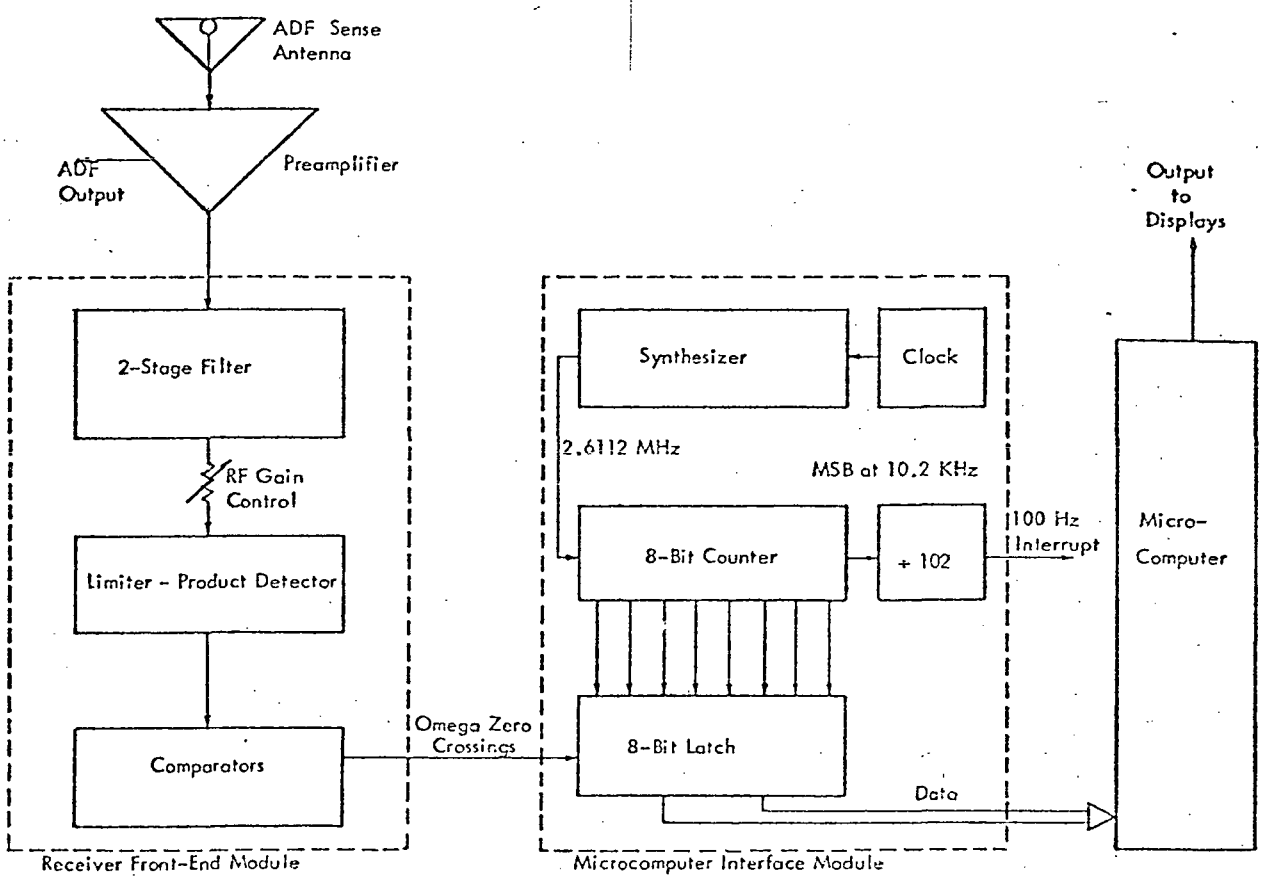

Figure 1. Summary Block Diagram - Microcomputer-Based Omega Receiver.

Function Generator Chip - A Japanese researcher[3] has developed a chip which will generate a Fourier series and a ROM will provide the parameters for the series. The skywave correction would be expressed as a variable of time having two periods, one daily and one yearly. This method has promise, but too many loose ends for present use. We continue to watch this work with interest for the future.

Skywave Memory - The skywave error from a station varies relatively slowly at any location over a short period of time (2-6 hours) excluding transition periods. Since airport Omega coordinates are known, the difference between the receiver calculated coordinates and the airport's published coordinates will yield a skywave correction factor that should improve positional accuracy for short flights less than 200 miles. This is a simple and inexpensive scheme, but it has limited application and accuracy.[4]

Pierce's[5] and Swanson's[6] algorithms have been tested using sophisticated airborne minicomputer systems. It is felt their development for implementation on a microprocessor based system was too ambitious for the present study.

A compressed tabular storage format for the skywave correction based on the Coast Guard[7] propagation tables was chosen for initial implementation. Further development of the microcomputer system in this paper will reflect this choice to some extent.

A block diagram for the proposed OMEGA navigation processor is included as Figure 2. That which is enclosed by the dashed lines will be implemented by the microprocessor in software and the boxes outside the dashed lines are input or output (hardware) devices. This software approach to navigation processing yields great flexibility and provides for the addition of air data inputs by simply adding additional I/O parts and software rather than a hardware redesign.

III. HARDWARE DESIGN

The hardware design of this system depends primarily upon the microcomputer system selected and secondarily upon the periphials it must control. The Motorola M6800 microprocessor family was chosen based on many factors such as: ease of I/O scheme, good instruction capability, 8-bit word size, single +5 volt power supply requirement, and documentation. The periphials are the Ohio University OMEGA sensor processor, a Kennedy digital incremental tape recorder, a sixteen-key keyboard and a Course Deviation Indicator driven through a Digital-to-Analog converter.

A detailed explanation of the M6800 system is beyond the scope of this paper, but a minimum system configuration is shown in Figure 3. The M6800 has an 8-bit bi-directional data bus and a 16-bit address bus (64K). There are 72 instructions of varying length with two accumulators for program usage. Input/output operations are simplified by M6820 Periphial Interface Adaptors (PIA) which may be accessed

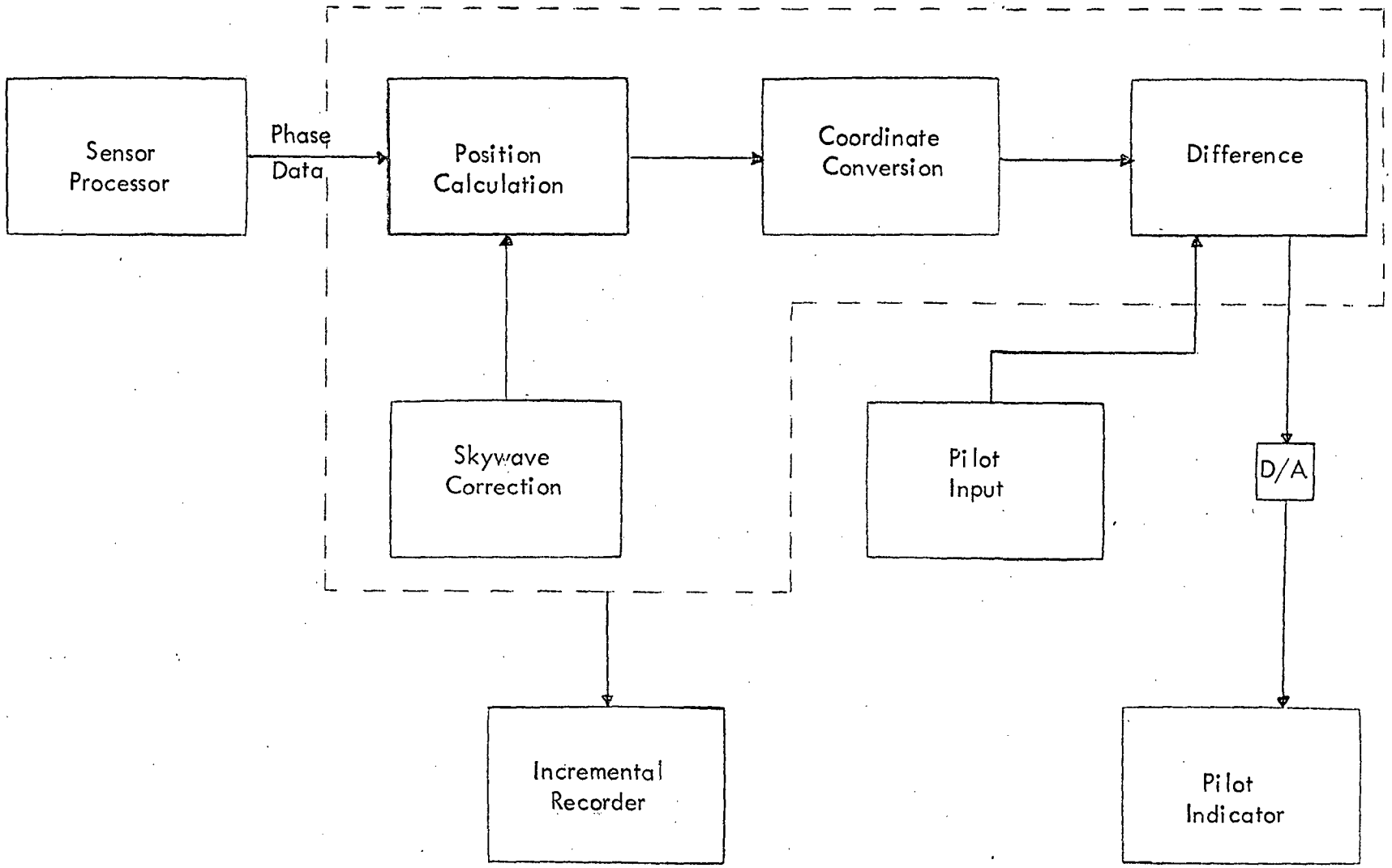

Figure 2. Omega Navigation Processor - Block Diagram.

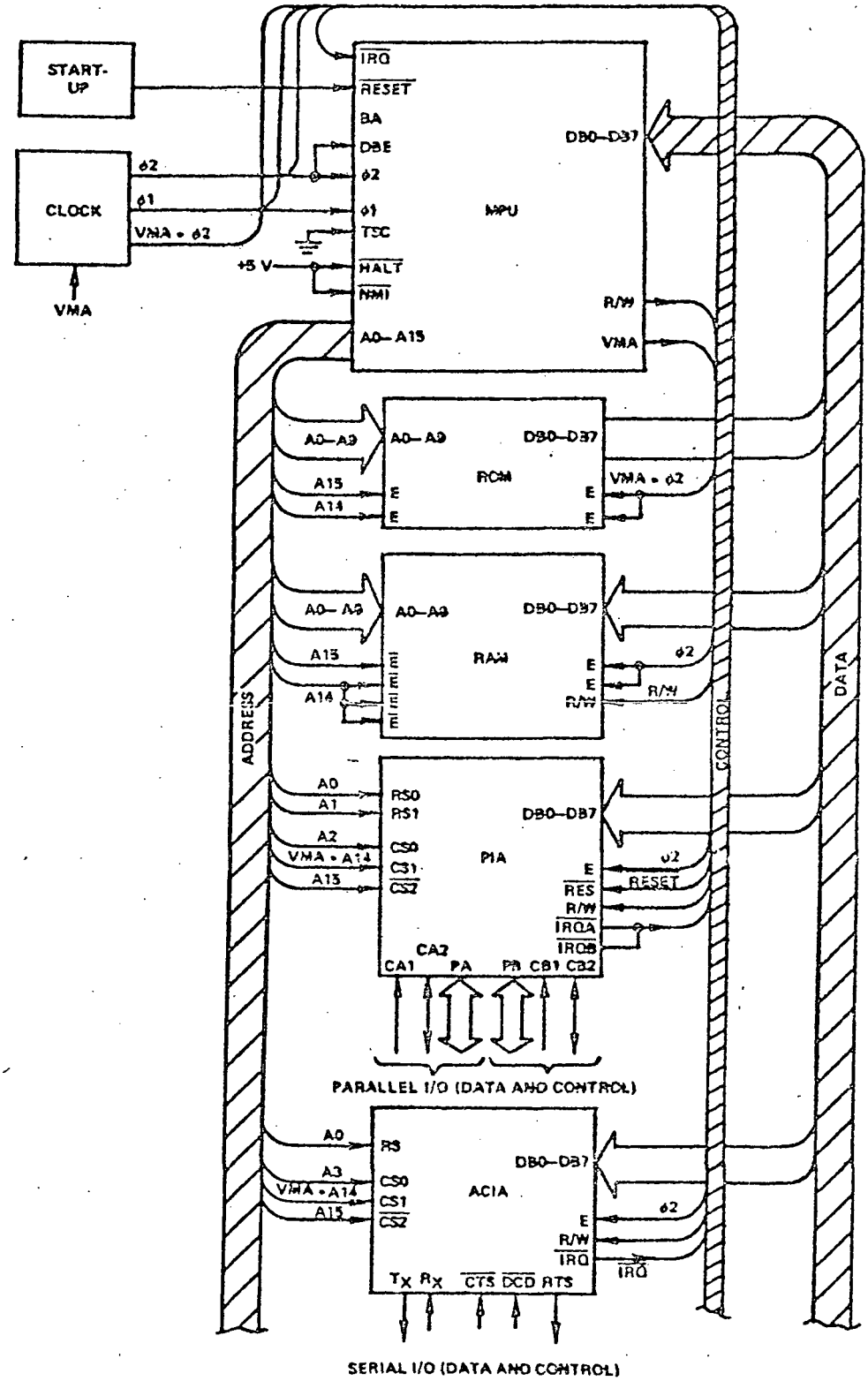

Figure 3. MPU Minimum System.

by a memory instruction directed to their location in memory. The PIA's are inherently parallel while the M6850 Asynchronous Communications Interface Adapter (ACIA) handle the serial interfaces. All family devices can drive 1 TTL load and up to 7 family devices. Inputs are TTL compatible. The microprocessor has Tri-State (TM) controls on most lines.

The microcomputer system configuration for the navigation processor will change somewhat from the development stage through final model. The software and skywave correction tables will be loaded into random-access memory (RAM) (4K-8K) in the beginning stages. Later progammable ROM (PROM) or mask ROM should be used. Erasable PROM's could be reprogrammed with updated skywave correction tables as needed.

Once the basic microcomputer system is configured, the Periphial Interface Adapters can be decoded to a convenient location in memory. Each PIA requires 8 data bits, 3 chip select and 2 register select lines, and 4 control lines from the main system bus. In addition, the PIA provides two 8-bit I/O channels with two control lines for each I/O channel. A diagram of this interface is shown in Figure 4.

As of this date, the receiver digital output interface has been specified but the details have not been published (NASA Contract NAS1-14124). A report on this is due in late summer of 1976. The pilot indicator interface has not yet been determined.

The Kennedy incremental recorder needs 8-bits of data valid during a 50 to 100 microsecond write pulse. The PIA has two periphial control lines, one of which can be used as a READY line (CB1) by the Kennedy. The other control line (CB2) can be programmed to provide a pulse when the B side of the PIA is written into. This in turn triggers a 555 to provide a 75 microsecond write pulse. By proper programming the READY line can be activated by the falling edge of the write pulse. The hardware required is, therefore, minimal, but the software requirements are more complex. Figure 5 is a diagram of both the Kennedy and keyboard interfaces.

The keyboard interface is, once again, simple in hardware and complex in software. The keyboard selected for use has 16 keys arranged in a 4 x 4 matrix. The four row and four column lines are connected directly to the A data side of a PIA. The column lines are also NANDed together (1/2-7420) to provide a KEYPRESS signal to the CA1 input of its PIA. The CA2 line need not be used at all. The program to read the data is unusual and will be discussed in the next section.

Since the development of the M6800 system, the NASA project has obtained a JOLT system, based on the MAI 6502 processor chip. This system utilizes PIA's for I/O operations also; therefore, the hardware interface should be directly compatible, but any software development might require modification. For development, the JOLT system should be interfaced to a teletype for the programming of RAM and diagnostic monitoring. (Program storage via paper tape.)

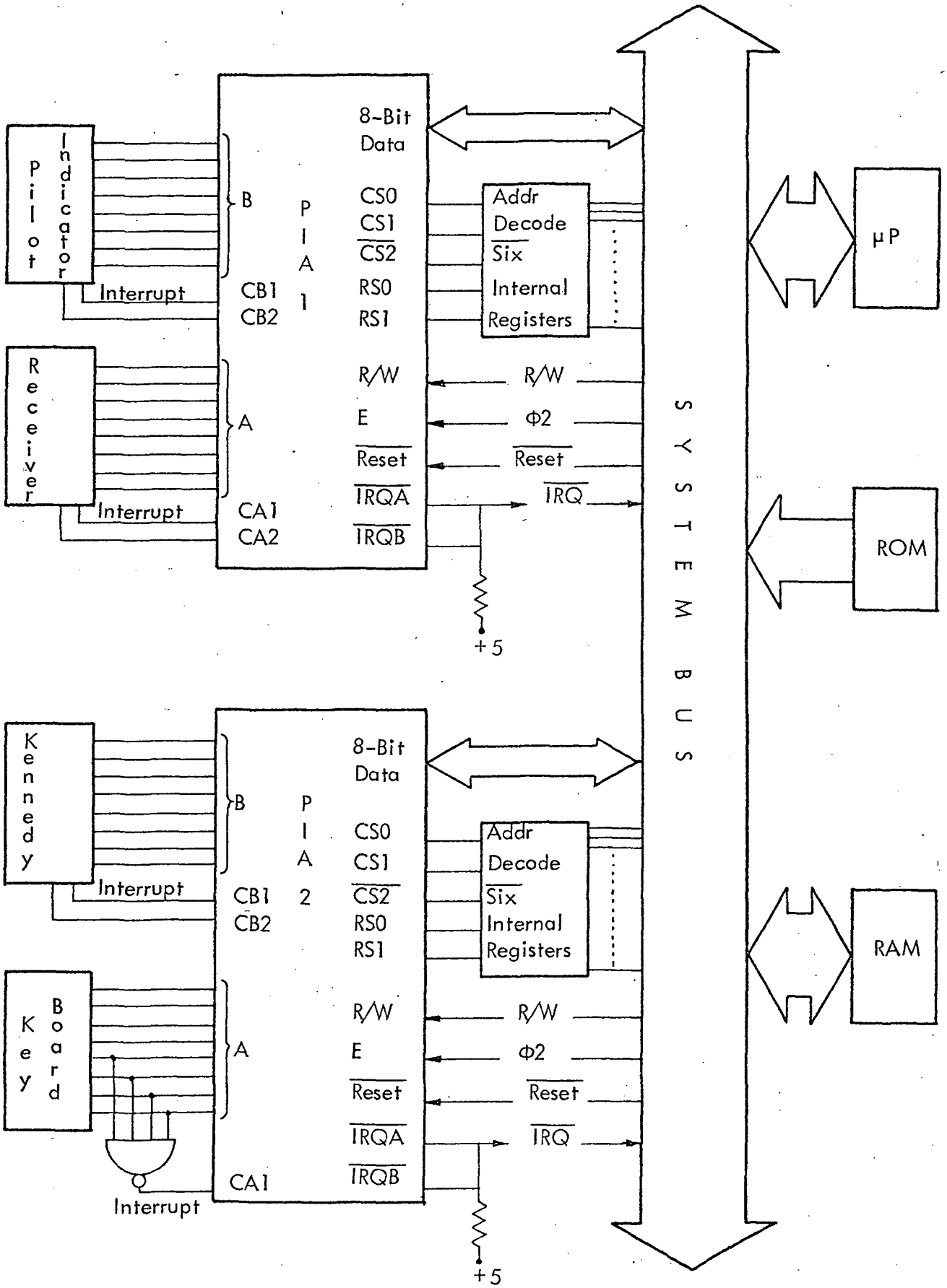

Figure 4. Omega Navigation Interface.

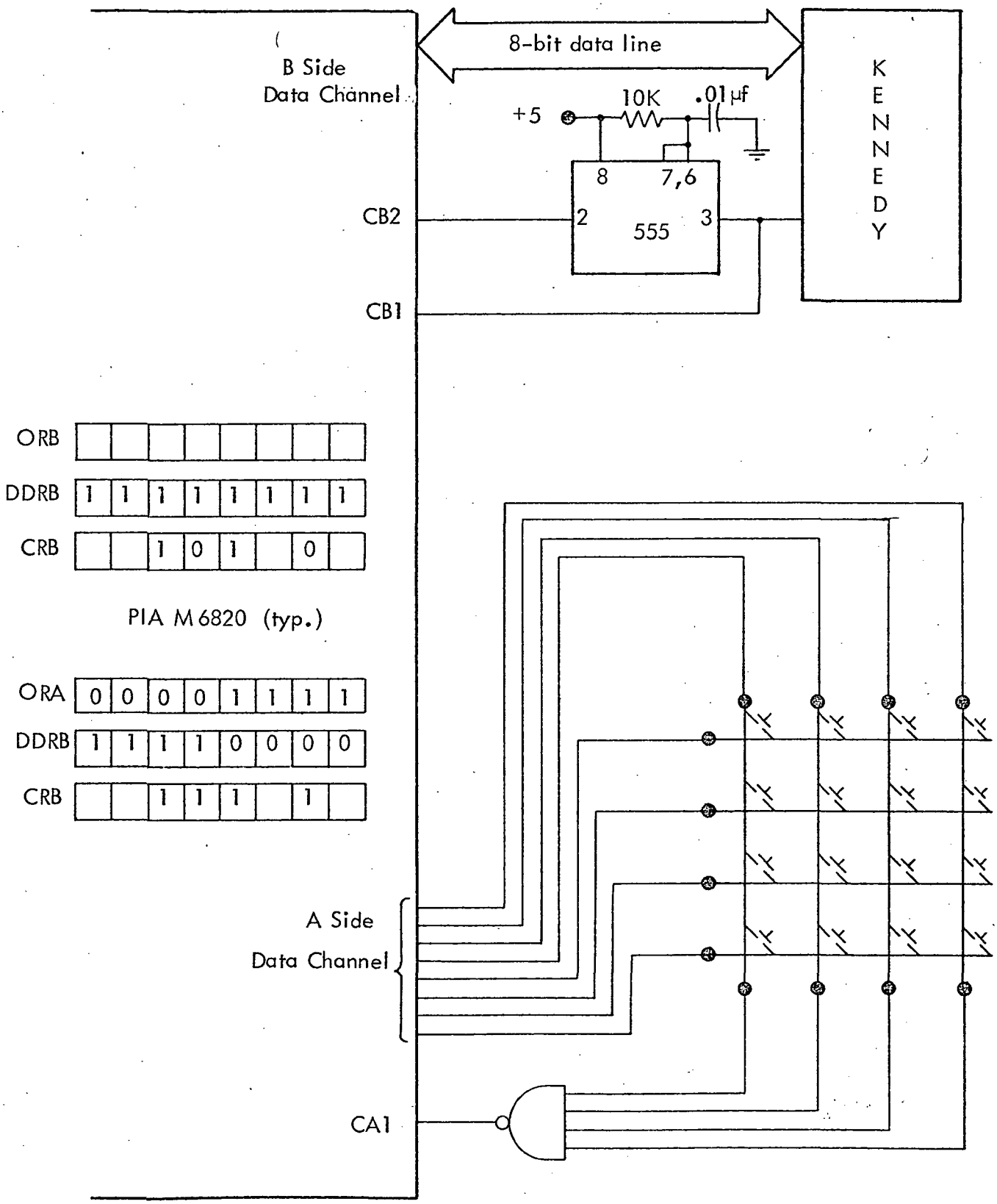

Figure 5. Kennedy and Keyboard Interface.

IV. SOFTWARE CONSIDERATIONS

A microprocessor cannot function without a controlling program stored in memory. That control program, along with interrelated subprograms, define what tasks the processor will do. There must be a subprogram for each input and output device. The main program will take the formatted inputs of OMEGA phase and pilot inputs, calculate the present position and/or heading, then transfer these values to the output subroutines.

The actual code for the programs will depend on the processor used, but the program concepts are the same. The main program algorithms are currently under investigation by Dr. Robert W. Lilley of the Avionics Engineering Center, and the I/O subprograms have been given some consideration by this author.

The pilot indicator output subroutine is as simple as writing into a PIA when a new value is calculated. The D/A converter can continuously read the current value. No control lines are needed.

The sensor processor interface will provide 6-bit phase data for any of four selected OMEGA stations. Service pulses for generating a "data valid" signal to the CA1 are also available on the receiver processor back plane. This signal will cause an interrupt and the main program will transfer control to the sensor processor input program to read the new phase data.

At the present time, the OMEGA signal can be supplied as filtered 6-bit phase for each time-slot or as 6-bit L.O.P. (Line-of-Position) output for any two station pairs selected from the receiver sensor processor. The 6-bit L.O.P. output is a single multiplexed reading alternately on two station pairs. A program subroutine in the microprocessor will be necessary to use either the L.O.P. data in this form or to perform a subtraction on the selected 6-bit phase data prior to using it for navigation computations.

The Kennedy incremental recorder output subprogram must determine if the Kennedy is ready to accept new data (not currently writing), write the data into the PIA output register B, then trigger the 555 for a write pulse. To pursue a more detailed description of the output software the PIA internal registers programming must be outlined. First, the B side Data Direction Register must be loaded with all '1's to define the B side of the PIA as an output port. This means that Output Register B will always accept data from the microprocessor and provide data to the Kennedy. Certain bit positions within Control Register B will be initialized next, and other control bits will be utilized throughout the program. A description of the control bit designations follows:

CB1 Controls
- CRB-0 program changeable
  - '0' $\overline{IRQB}$ disabled
  - '1' $\overline{IRQB}$ enable data ready to write
- CRB-1 initialize to '0'; CB1 active on falling edge
- CRB-2 '0' when initializing DDRB to all '1'
  '1' normally to allow data flow

CB2 Controls

- CRB-3 initialize to '1'
- CRB-4 initialize to '0'
- CRB-5 initialize to '1'

} programs CB2 to give an output low pulse after a B side write by microprocessor.

CRB-6 not used when CB2 used an output
CRB-7 set by CB1, reset by microprocessor read of output register B.

The program should initialize DDRB then set up Control Register bit positions one through five. When the program has data ready for the Kennedy it should change CRB-0 from '0' to '1' and examine CRB-7. If CRB-7 is a '1' the data should be written into Output Register B. That write operation triggers CB2 and the 555 outputs a write pulse. CRB-7 should be reset by reading Output Register B. The main program can now turn to its next task.

The program to read data from a non-encoded keyboard can be adapted from an example in the Motorola M6800 Microprocessor Applications Manual, Section 5-1.1.2. This is a clever scheme to output '0's in the column lines and sense a '0' in the row line with a key closure. The microprocessor then outputs '0's on the row lines and reads the column lines to determine the exact one of sixteen keys that is closed. The so-derived data word (e.g. 1011 1101) must be matched in a table to generate the proper microprocessor usable code.

To accomplish the above sequence the A side Registers must be properly manipulated. For an interrupt to reach the microprocessor CRA must be initialized to all 1's. After an interrupt is acknowledged, CRA-2 should be reset '0', then DDRA can be loaded with 1111 0000 and then CRA-2 is set to '1' so Output Register A can be loaded with 0000 1111. The keyboard rows can now be read. Switch the contents of DDRA and ORA to read the columns, then begin the table search.

This concludes discussion of the software requirements. Obviously the system is incompletely specified at this stage, but a detailed input and output sequence have been illustrated and the basic requirements for further development are noted.

V. CONCLUSION

The hardware interface designed in this project will operate with either the M6800 system or the JOLT system. However, the actual software codes will vary depending on the microprocessor. This report begins with our early discussions concerning the OMEGA navigation processor approximately one year ago and brings us to the current status and possible extensions of this work. The hardware design of the PIA circuitry is complete, and the parts are ready to be assembled. The software is outlined and requires further development. The microprocessor system is built and is currently being debugged. To implement a working navigation system, these tasks must be completed.

VI. REFERENCES

[1] U. S. Naval Oceanographic Office, "Omega Tables - Area 11, North America, Pair B-D", H.O. Publishers, No. 224 (111) B-D, 1971.

[2] Josephy, N. H., and Kasper, J. F., "A Polynomial Technique for Small-Computer Skywave Correction Implementation", Proceedings of the First Omega Symposium, pp. 56-62, November 9-11, 1971.

[3] Shimomura, Naonobu, in a letter to Mr. R. W. Burhans, Avionics Engineering Center, Ohio University, Athens, Ohio, received February 25, 1975.

[4] Matzke, David E., DBA Systems, Inc., in a letter to the Avionics Engineering Center, Ohio University, Athens, Ohio, dated January 2, 1975.

[5] Baltzer, O. J., and Fraser, E. C., "Omega Navigation Systems Specifications and Performance Measurement", Proceedings of the Second Omega Symposium, pp. 42-45, November 5-7, 1974.

[6] Josephy and Kasper, op. cit.

[7] U. S. Naval Oceanographic Office, op. cit.